Debian: empty /etc, empty /var and factory reset

The path to stateless systems

Marco d'Itri
<md@linux.it>
@md@linux.it

Seeweb s.r.l. + Debian

All Systems Go! 2024 - September 25, 2024

Ephemeral systems and factory reset

Linux factory reset

Support restoring a system to the default configuration by deleting `/etc/` and `/var/`.

Ephemeral systems

If `/etc/` and `/var/` are on a tmpfs then the system state will be reset at the next boot. More generally, everything but `/usr/` can be a tmpfs: even `/`.

The obvious use cases:

- Appliances
- Containers and other large fleets
- Generic desktops

The state of stateless systems in Debian

- What works?
- What is missing?
- What can we hack to make it work right now?

The good news:

Surprisingly few workarounds are needed for a minimal system.

This presentation discusses Debian systems, but most concepts are generally applicable to other Linux distribution.

How to experiment

We can easily use systemd-nspawn to boot an ephemeral clone of the current system, with some minor adjustments.

Copy some files, to be restored by `systemd-tmpfiles`:

```
mkdir -p /usr/share/factory/etc/
cp -a /etc/protocols /etc/rpc /etc/services /etc/ethertypes /etc/nsswitch.conf \
  /etc/pam.d /etc/security /etc/alternatives /usr/share/factory/etc/
vi /usr/lib/tmpfiles.d/factory-etc.conf
```

Create an empty container:

```
mkdir /var/lib/machines/empty/
ln -s usr/bin usr/sbin usr/lib usr/lib64 /var/lib/machines/empty/
```

Start the container:

```
SYSTEMD_NSPAWN_CHECK_OS_RELEASE=0 \
systemd-nspawn --private-network --network-veth -b -D /var/lib/machines/empty/ \
  --bind-ro=/usr --tmpfs=/etc/ --tmpfs=/var/ --tmpfs=/tmp/
```

(Is `--tmpfs=/` actually supposed to work?)

Using `systemd-tmpfiles` to restore the missing parts

It implements a generic mechanism to restore on first boot some files from `/usr/share/factory/`.

They can be listed e.g. in `/usr/lib/tmpfiles.d/factory-etc.conf`:

```
L /etc/ethertypes
L /etc/protocols
L /etc/rpc
L /etc/services
C /etc/nsswitch.conf

C+ /etc/pam.d/
C+ /etc/security/
```

Notable workarounds for `/etc/`: PAM

`/etc/security/`

It contains the configuration of PAM modules, also set by systemd (`/etc/security/limits.d/`). There is no `/usr/` equivalent.

`/etc/pam.d/`

These files are slowly being moved[1] to `/usr/lib/pam.d/`, except for...

`/etc/pam.d/common-*` (Debianism, but a good one)

These files are the automatically modified PAM configurations, included by the others. We can run `pam-auth-update --force` at boot to rebuild them.

[1] Maybe not in Red Hat, where they are locally modified? For Debian, wait for DH compatibility level 14.

Notable workarounds for `/etc/`: `update-alternatives`

`update-alternatives(1)`

Manages user-configurable symlinks chains like:
`/usr/bin/awk` ⇒ `/etc/alternatives/awk` ⇒ `/usr/bin/mawk`

It is useful to choose a system default between `gawk` or `mawk`, or among different versions of `gcc`.
This feature is as old as Debian and widely beloved, so I do not believe that it can be removed just to make factory reset easier.

The default links can be automatically regenerated, but not if they have been configured manually: `/etc/alternatives/` *is* the configuration.

Notable workarounds for `/etc/`: the libc networking

Protocol numbers databases (e.g. `getprotobyname(3)`)

`/etc/ethertypes`, `/etc/protocols`, `/etc/rpc` and `/etc/services` need to be restored from `/usr/`.

`/etc/nsswitch.conf`

It is edited by the maintainer scripts of the NSS plugins, no solution yet!

`/etc/hosts`

It can be replaced by `libnss-myhostname` to resolve `localhost`, but it needs to be enabled in `/etc/nsswitch.conf`!

Notable workarounds: dpkg

The dpkg database can just be moved to /usr/:

```
mv /var/lib/dpkg /usr/lib/dpkg-state
ln -s /usr/lib/dpkg-state /var/lib/dpkg
```

/usr/ is immutable, so the installed packages cannot change anyway.

But I am not sure of how to handle /var/lib/dpkg/alternatives/.
(Current strategy: copy it to /var/ and then bind-mount it back.)

Also, the content of /etc/dpkg/origins/ should be moved to /usr/ (see #1041210).

Find non-compliant packages

If you want to help, have a look at what files are installed in `/var/`:

```
for file in $(egrep -h ^/var/ /var/lib/dpkg/info/*.list | sort -u); do
  [ ! -L "$file" -a -f "$file" ] && echo "$file"
done | less
```

But some are false positives: e.g. `units_cur(1)` can rebuild `/var/lib/units/`.

And some other files can be ignored, e.g. `/var/lib/*/README*`.

Towards a Debian policy for stateless systems?

A first goal

It would be reasonable to have support for factory reset from priority required packages and other cooperating packages. An empty `/var/` is **much** closer than `/etc/`.

It is not obvious to me that it would be useful to mandate support from all packages.

Encourage (mandate?) more declarative packaging:

- Use `systemd-sysusers` to create users and groups.
- Use `{Runtime,State,Cache,Logs,Configuration}Directory=` and `systemd-tmpfiles` to create the directories for the daemons.

(These can be used in parallel with the old methods, if desired!)

Interim workarounds?

We could cheat and take the `usrmerge` approach...

The `factory-reset-support` package could:

- Use `tmpfiles.d` files to create the `/etc/` and `/var/` directories which appear in the `dpkg` database but are not also created on demand by systemd.
- Use `sysusers.d` files to create the users and groups which are only added by the maintainer scripts.
- Use `/usr/share/factory/` for each missing file until a proper solution will have been implemented.
- Install some units to rebuild things on first boot.

I have an unreleased but working proof of concept.

Any questions?

https://www.linux.it/~md/text/factoryreset-asg2024.pdf

(Google ... Marco d'Itri ... I'm feeling lucky)